Lab 1: Getting Started with Node.js

In this lab, you will become familiar with JavaScript by putting into practice what you have seen in the last lectures.

0. Preparation and warm-up exercise

Before starting, make sure that Node.js has been installed on your computer.

Then, create a function that, given an array of strings, for each string computes and prints a new one composed by the first two and last two characters. If the word is shorter than two characters, the function will print an empty string. Otherwise, if the word is two characters long, the function prints the same character twice.

Examples: '`spring`' yields '`spng`'; '`it`' yields '`itit`'; '`cat`' yields '`caat`'.

Call the function with a variety of strings and check the result's correctness.

1. Create a Film Library

In this exercise, you will implement a simple application to track the films that a person wants to watch and the ones they have already watched. Each film is represented by the following fields:

- A unique numerical **id** (mandatory)
- A **title** (mandatory)
- A Boolean value to represent whether the film is among the person's **favorites** (default value: false)
- A date corresponding to the **date** when the person **watched the film** (optional)
- A numerical value between 1 and 5 to represent the **rating** that the person has given to the film after watching it (optional)
- A numerical id representing the person (mandatory, default to 1).

Firstly, implement a constructor function to create **Film** objects.

Secondly, implement a constructor function to create a **FilmLibrary**, an object containing an array of Films.

Then, implement the **`addNewFilm`** method, which adds a new **Film** object, passed as parameter, to the **FilmLibrary**. **Populate** the **FilmLibrary** using this method.

To conclude, print in the console the entire list of Films stored in the FilmLibrary, with all their fields.

For instance, you can take inspiration from the following list:

```
Id: 1, Title: Pulp Fiction, Favorite: true, Watch date: March 10, 2024, Score: 5, User: 1
Id: 2, Title: 21 Grams, Favorite: true, Watch date: March 17, 2024, Score: 4, User: 1
Id: 3, Title: Star Wars, Favorite: false, Watch date: null, Score: 0, User: 1
Id: 4, Title: Matrix, Favorite: false, Watch date: null, Score: 0, User: 1
Id: 5, Title: Shrek, Favorite: false, Watch date: March 21, 2024, Score: 3, User: 1
```

Hint: you may use the day.js library to create and handle the dates.

Hint: To implement the required functionalities described above you may use the functional programming paradigm to manipulate the array of films.

2. Add functionalities to the Film Library

In this exercise you will add a set of methods to the **FilmLibrary** object, namely:

- **`sortByDate`**: returns a new array containing the Films within the **FilmLibrary** instance sorted in ascending order of the watch date. The movies that the user has not already watched should be put at the end. For example, after the sorting, the **FilmLibrary** shown in the previous exercise would look like:

```
***** List of films *****
Id: 1, Title: Pulp Fiction, Favorite: true, Watch date: March 10, 2024, Score: 5, User: 1
Id: 2, Title: 21 Grams, Favorite: true, Watch date: March 17, 2024, Score: 4, User: 1
Id: 5, Title: Shrek, Favorite: false, Watch date: March 21, 2024, Score: 3, User: 1
Id: 3, Title: Star Wars, Favorite: false, Watch date: null, Score: 0, User: 1
Id: 4, Title: Matrix, Favorite: false, Watch date: null, Score: 0, User: 1
```

- **`deleteFilm`**: deletes a **Film** from the **FilmLibrary** based on an Id received by parameter.

- **`resetWatchedFilms`**: deletes the Watch date of all the **Films** in the **FilmLibrary**.

- **`getRated`**: selects the films that do have a defined score. Only movies with an assigned score should be returned, ordered by decreasing score. After filtering the Films Library shown in exercise 1, the method should print:

```
***** Films filtered, only the rated ones *****
Id: 1, Title: Pulp Fiction, Favorite: true, Watch date: March 10, 2024, Score: 5
Id: 2, Title: 21 Grams, Favorite: true, Watch date: March 17, 2024, Score: 4
Id: 5, Title: Shrek, Favorite: false, Watch date: March 21, 2024, Score: 3
```

Hint: To implement the required functionalities described above you may use the functional programming paradigm to manipulate the array of films.

Finally, test the methods by invoking them over the **FilmLibrary** instance you created and populated in exercise 1.